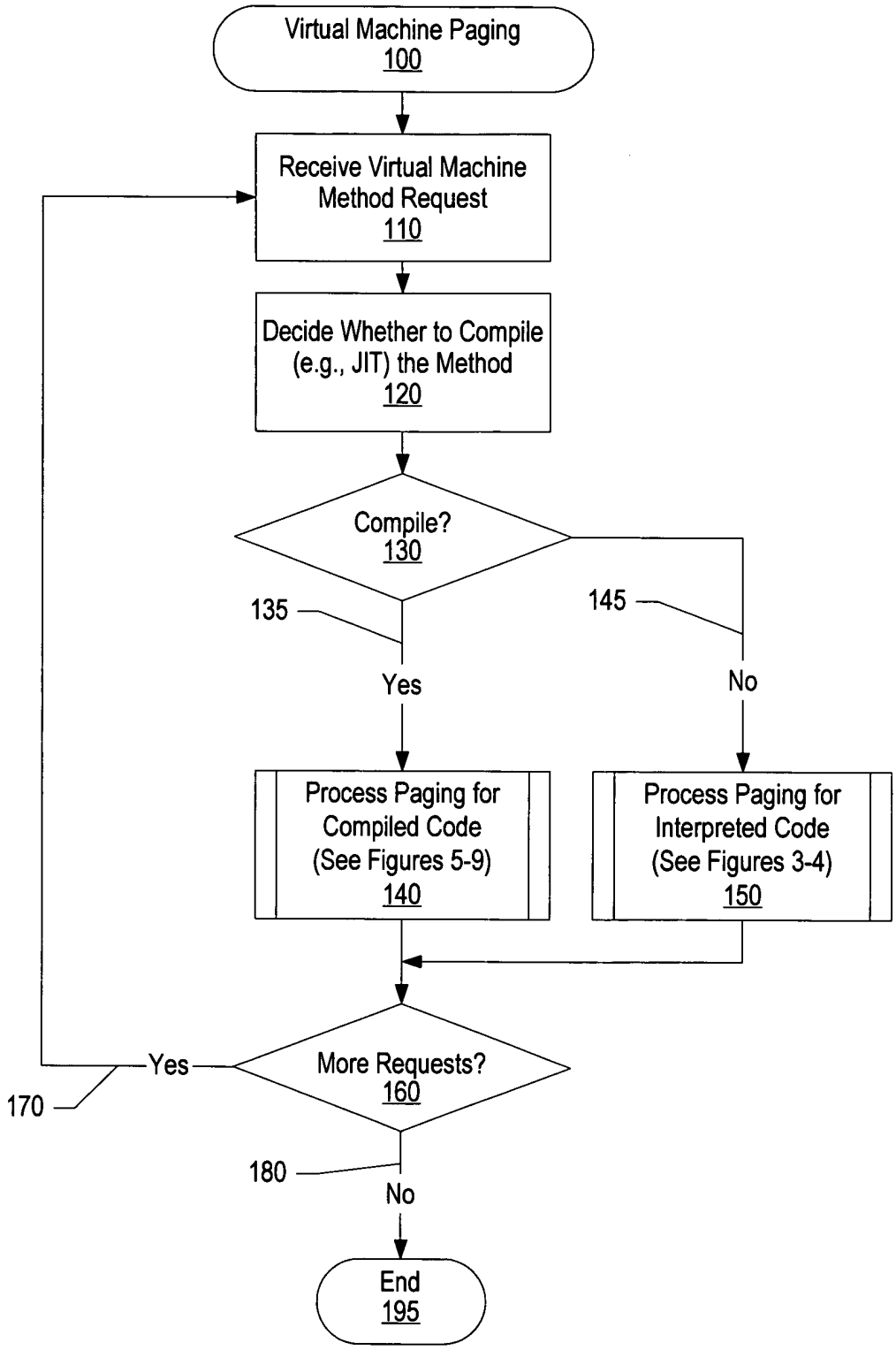
Virtual Machine Paging
100
Receive Virtual Machine Method Request
110
Decide Whether to Compile (e.g., JIT) the Method
120
Compile?
130
135
Yes
145
No
Process Paging for Compiled Code (See Figures 5-9)
140
Process Paging for Interpreted Code (See Figures 3-4)
150
More Requests?
160
Yes
170
180
No
End
195

Figure 1

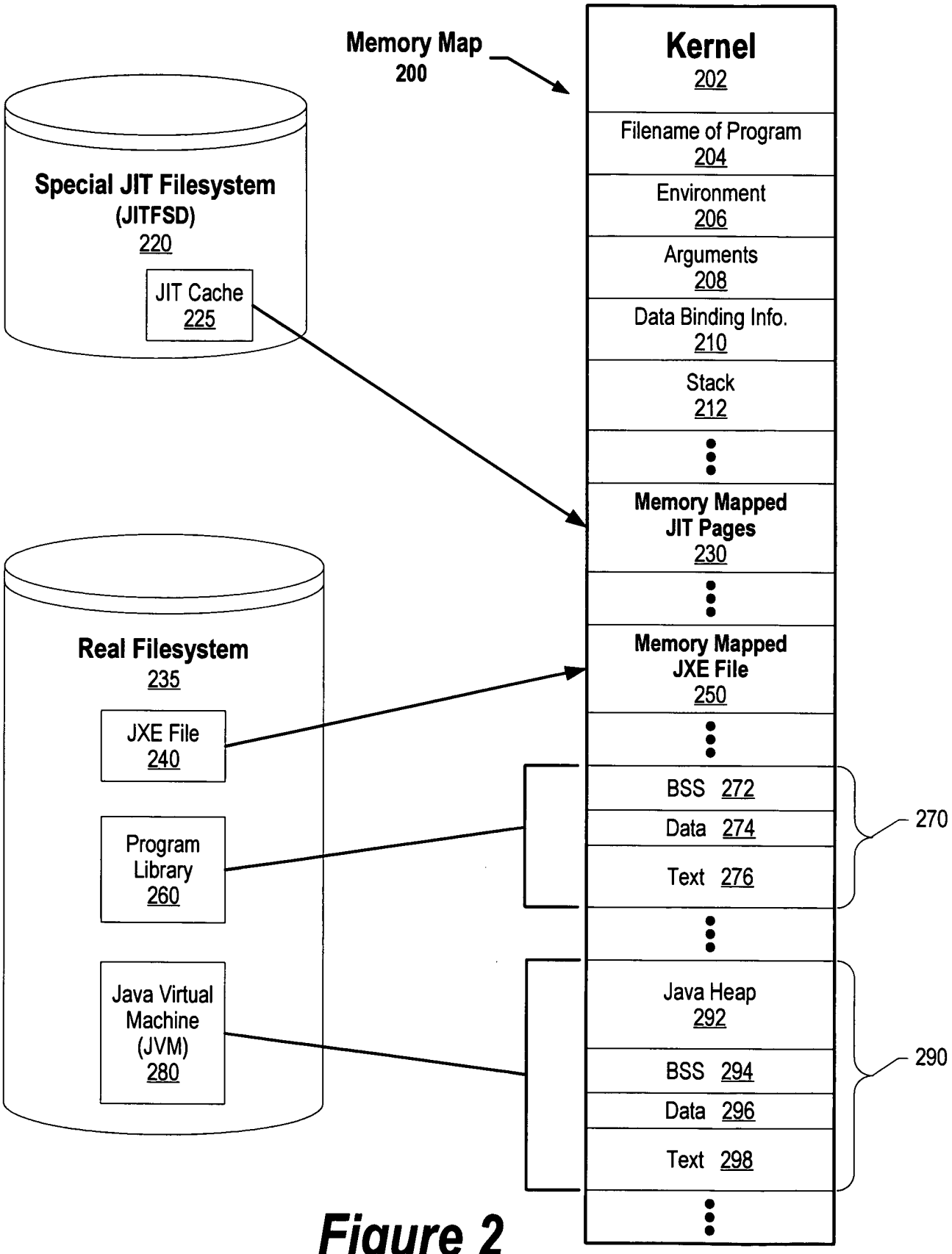
Memory Map
200
Kernel
202
Filename of Program
204
Environment
206
Arguments
208
Data Binding Info.
210
Stack
212
Memory Mapped
JIT Pages
230
Memory Mapped
JXE File
250
BSS 272
Data 274
Text 276
270
Java Heap
292
BSS 294
Data 296
Text 298
290
Special JIT Filesystem
(JITFSD)
220
JIT Cache
225
Real Filesystem
235
JXE File
240
Program
Library
260
Java Virtual
Machine
(JVM)
280

Figure 2

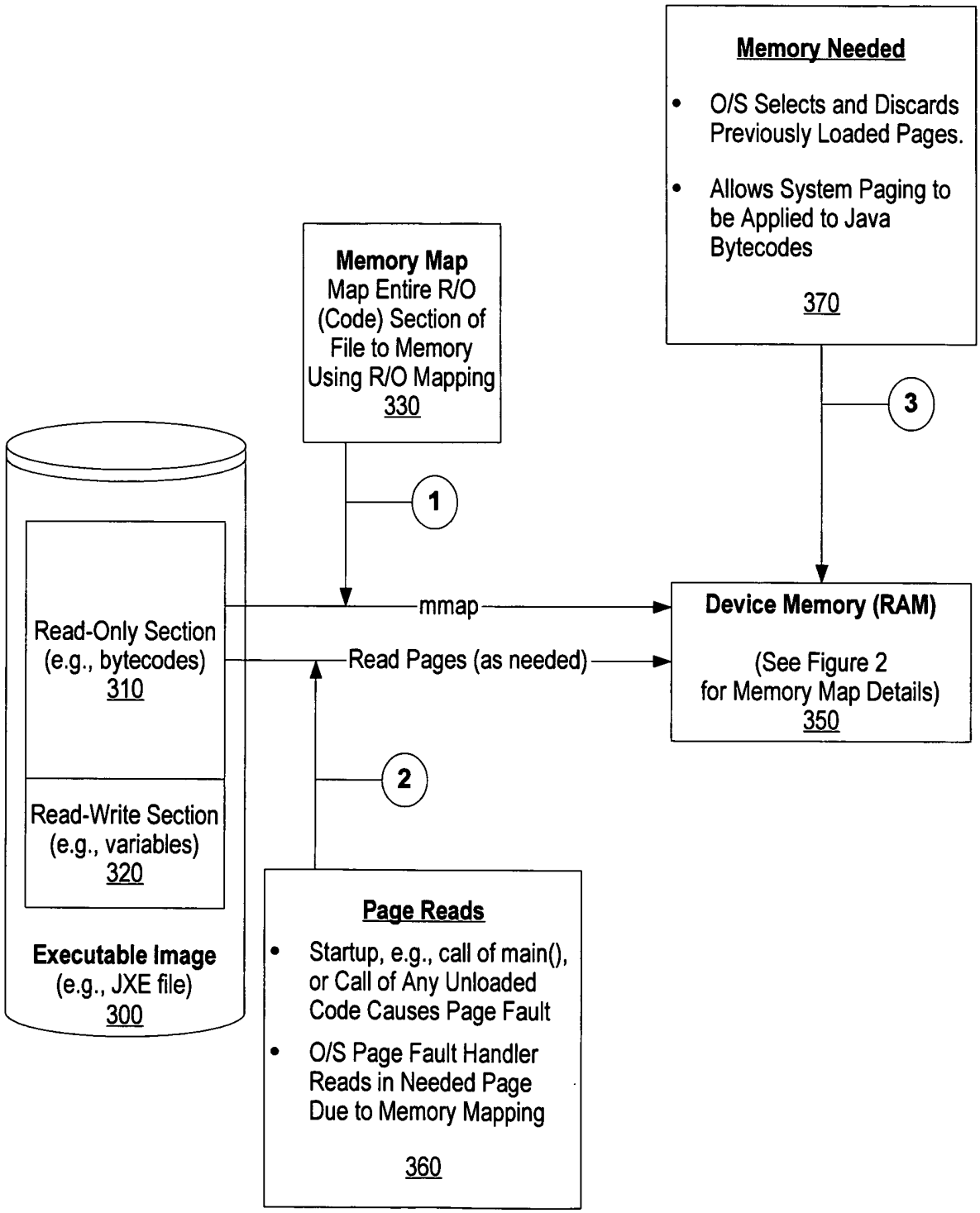
Memory Needed
• O/S Selects and Discards Previously Loaded Pages.
• Allows System Paging to be Applied to Java Bytecodes
370
Memory Map
Map Entire R/O (Code) Section of File to Memory Using R/O Mapping
330
3
1
Read-Only Section (e.g., bytecodes)
310
mmap
Device Memory (RAM)
(See Figure 2 for Memory Map Details)
350
Read Pages (as needed)
2
Read-Write Section (e.g., variables)
320
Executable Image (e.g., JXE file)
300
Page Reads
• Startup, e.g., call of main(), or Call of Any Unloaded Code Causes Page Fault
• O/S Page Fault Handler Reads in Needed Page Due to Memory Mapping
360

Figure 3

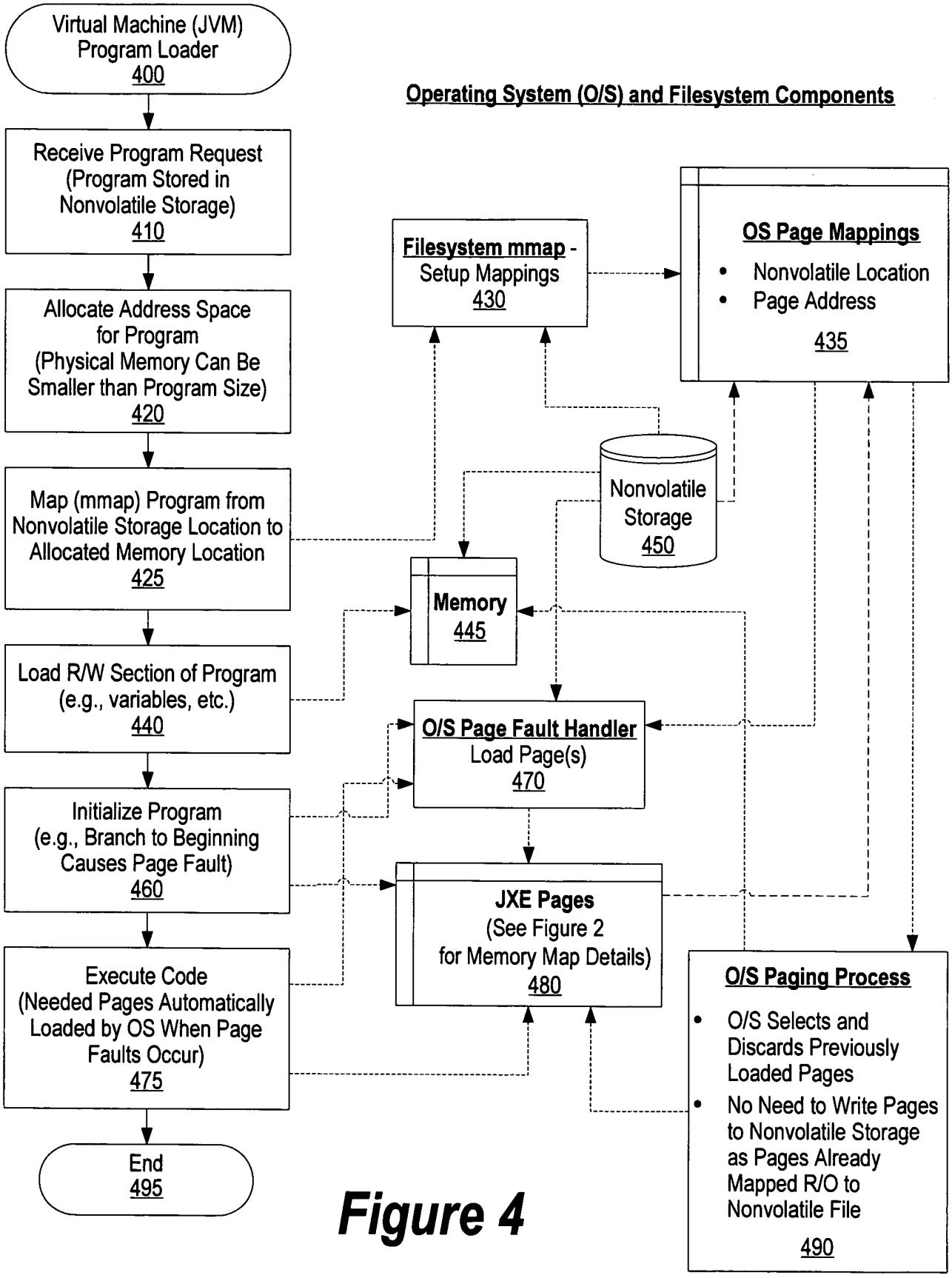
Virtual Machine (JVM) Program Loader 400
Receive Program Request (Program Stored in Nonvolatile Storage) 410
Allocate Address Space for Program (Physical Memory Can Be Smaller than Program Size) 420
Map (mmap) Program from Nonvolatile Storage Location to Allocated Memory Location 425
Load R/W Section of Program (e.g., variables, etc.) 440
Initialize Program (e.g., Branch to Beginning Causes Page Fault) 460
Execute Code (Needed Pages Automatically Loaded by OS When Page Faults Occur) 475
End 495
Operating System (O/S) and Filesystem Components
Filesystem mmap - Setup Mappings 430
OS Page Mappings
Nonvolatile Location
Page Address
435
Nonvolatile Storage 450
Memory 445
O/S Page Fault Handler
Load Page(s) 470
JXE Pages
(See Figure 2 for Memory Map Details) 480
O/S Paging Process
O/S Selects and Discards Previously Loaded Pages
No Need to Write Pages to Nonvolatile Storage as Pages Already Mapped R/O to Nonvolatile File
490

Figure 4

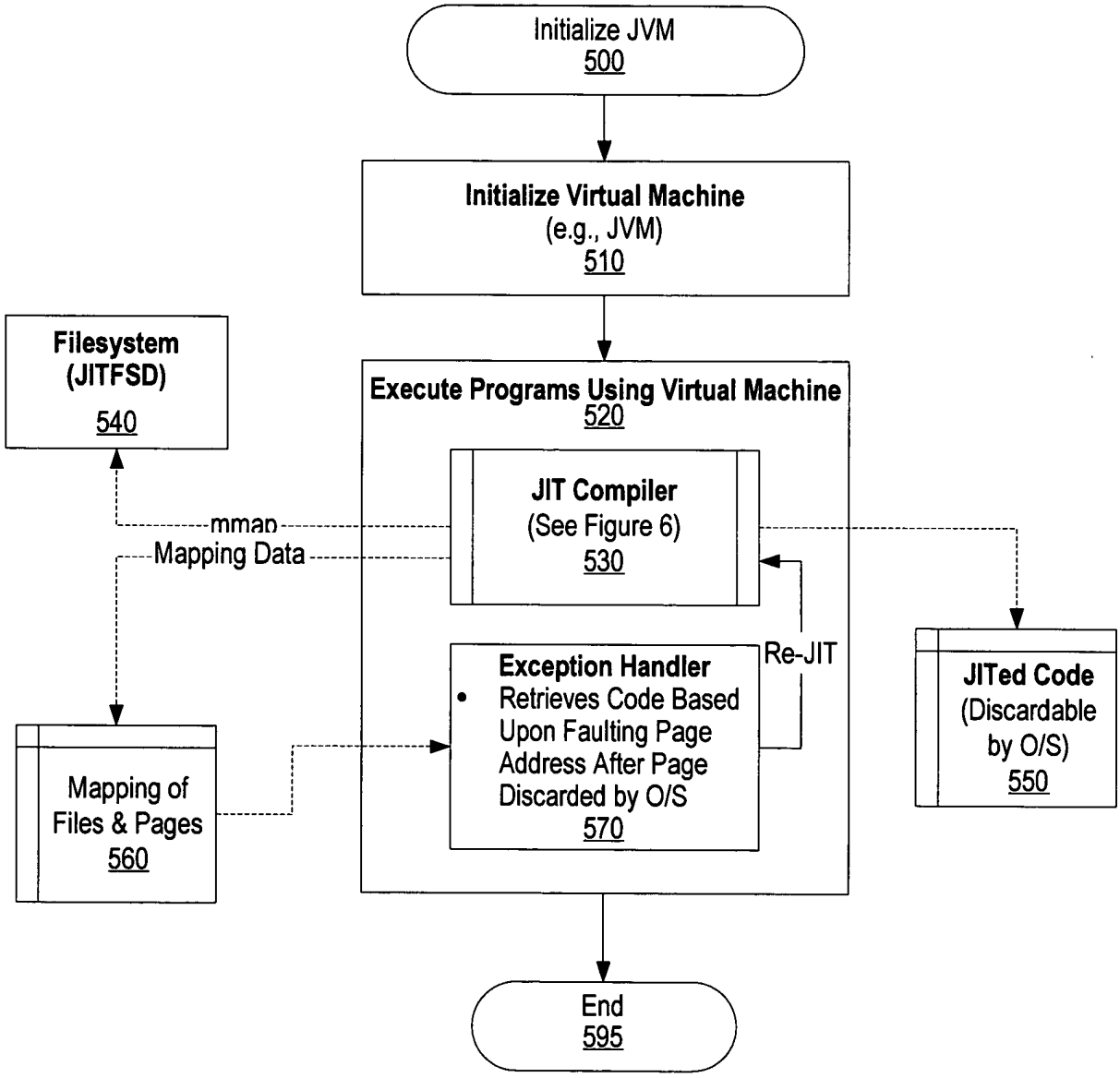
Initialize JVM
500
Initialize Virtual Machine
(e.g., JVM)
510
Filesystem
(JITFSD)
540
Execute Programs Using Virtual Machine
520
JIT Compiler
(See Figure 6)
530
mmap
Mapping Data
Exception Handler
• Retrieves Code Based Upon Faulting Page Address After Page Discarded by O/S
570
Re-JIT
JITed Code
(Discardable by O/S)
550
Mapping of Files & Pages
560
End
595

Figure 5

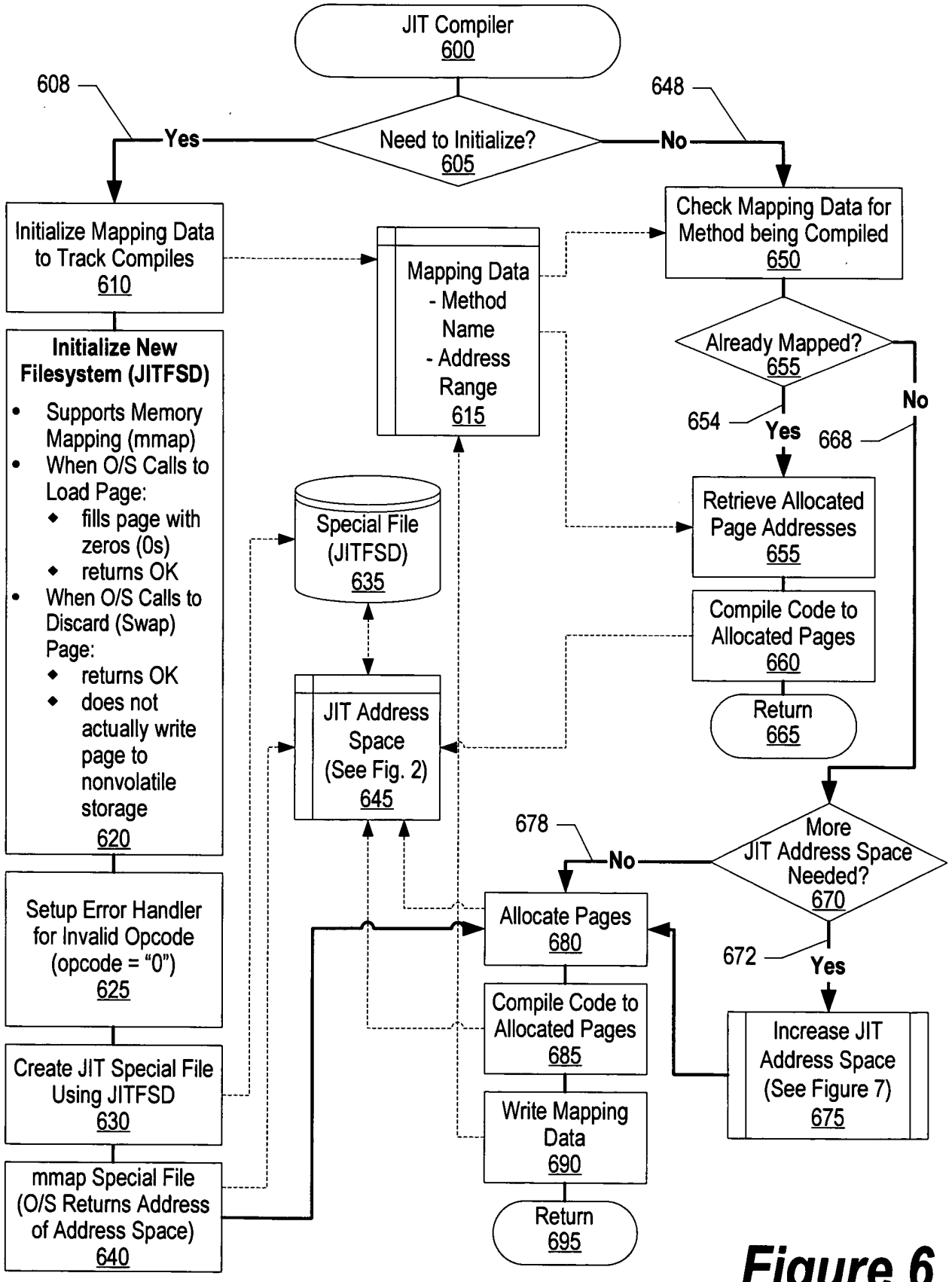
JIT Compiler
600
Need to Initialize?
605
608
Yes
648
No
Initialize Mapping Data to Track Compiles
610
Initialize New Filesystem (JITFSD)
• Supports Memory Mapping (mmap)
• When O/S Calls to Load Page:
◆ fills page with zeros (0s)
◆ returns OK
• When O/S Calls to Discard (Swap) Page:
◆ returns OK
◆ does not actually write page to nonvolatile storage
620
Setup Error Handler for Invalid Opcode (opcode = "0")
625
Create JIT Special File Using JITFSD
630
mmap Special File (O/S Returns Address of Address Space)
640
Mapping Data
- Method Name
- Address Range
615
Special File (JITFSD)
635
JIT Address Space (See Fig. 2)
645
Check Mapping Data for Method being Compiled
650
Already Mapped?
655
654
Yes
No
668
Retrieve Allocated Page Addresses
655
Compile Code to Allocated Pages
660
Return
665
More JIT Address Space Needed?
670
678
No
672
Yes
Allocate Pages
680
Compile Code to Allocated Pages
685
Write Mapping Data
690
Return
695
Increase JIT Address Space (See Figure 7)
675

Figure 6

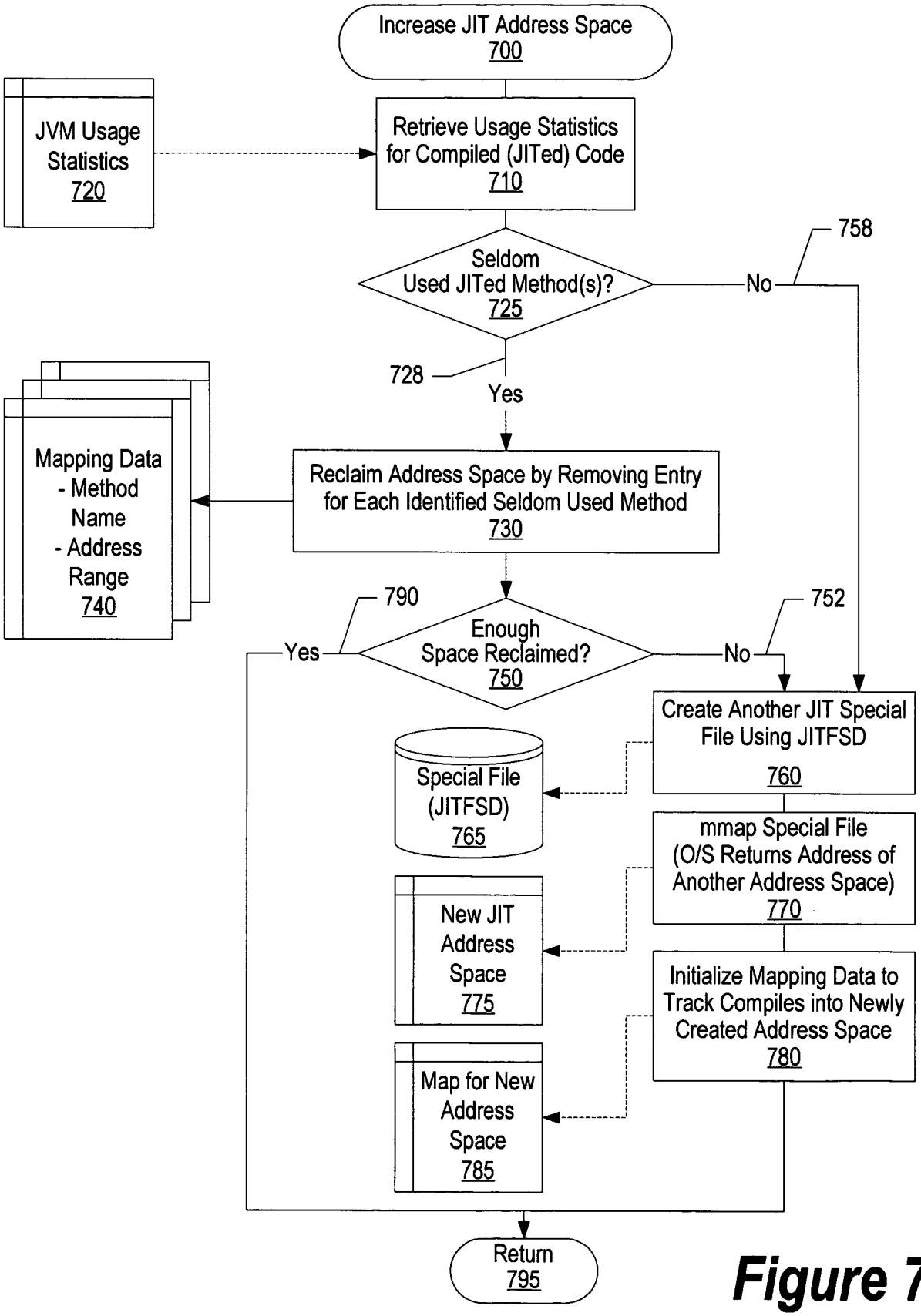
Increase JIT Address Space
700
JVM Usage Statistics
720
Retrieve Usage Statistics for Compiled (JITed) Code
710
Seldom Used JITed Method(s)?
725
758
No
728
Yes
Mapping Data
- Method Name
- Address Range
740
Reclaim Address Space by Removing Entry for Each Identified Seldom Used Method
730
790
Yes
Enough Space Reclaimed?
750
No
752
Create Another JIT Special File Using JITFSD
760
Special File (JITFSD)
765
mmap Special File (O/S Returns Address of Another Address Space)
770
New JIT Address Space
775
Initialize Mapping Data to Track Compiles into Newly Created Address Space
780
Map for New Address Space
785
Return
795

Figure 7

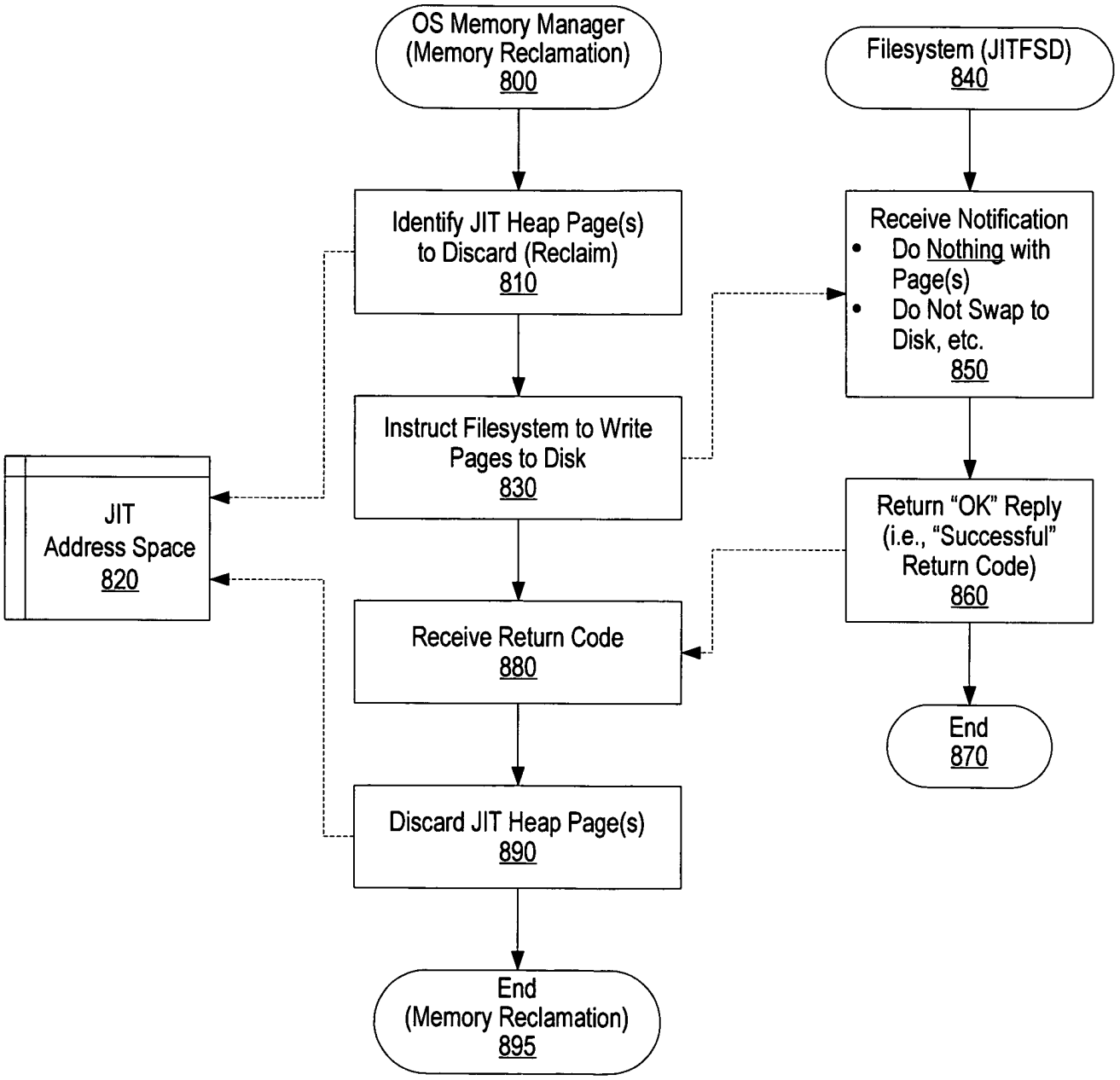
OS Memory Manager (Memory Reclamation) 800
Filesystem (JITFSD) 840
Identify JIT Heap Page(s) to Discard (Reclaim) 810
Receive Notification
• Do Nothing with Page(s)
• Do Not Swap to Disk, etc.
850
Instruct Filesystem to Write Pages to Disk 830
JIT Address Space 820
Return "OK" Reply (i.e., "Successful" Return Code) 860
Receive Return Code 880
End 870
Discard JIT Heap Page(s) 890
End (Memory Reclamation) 895

Figure 8

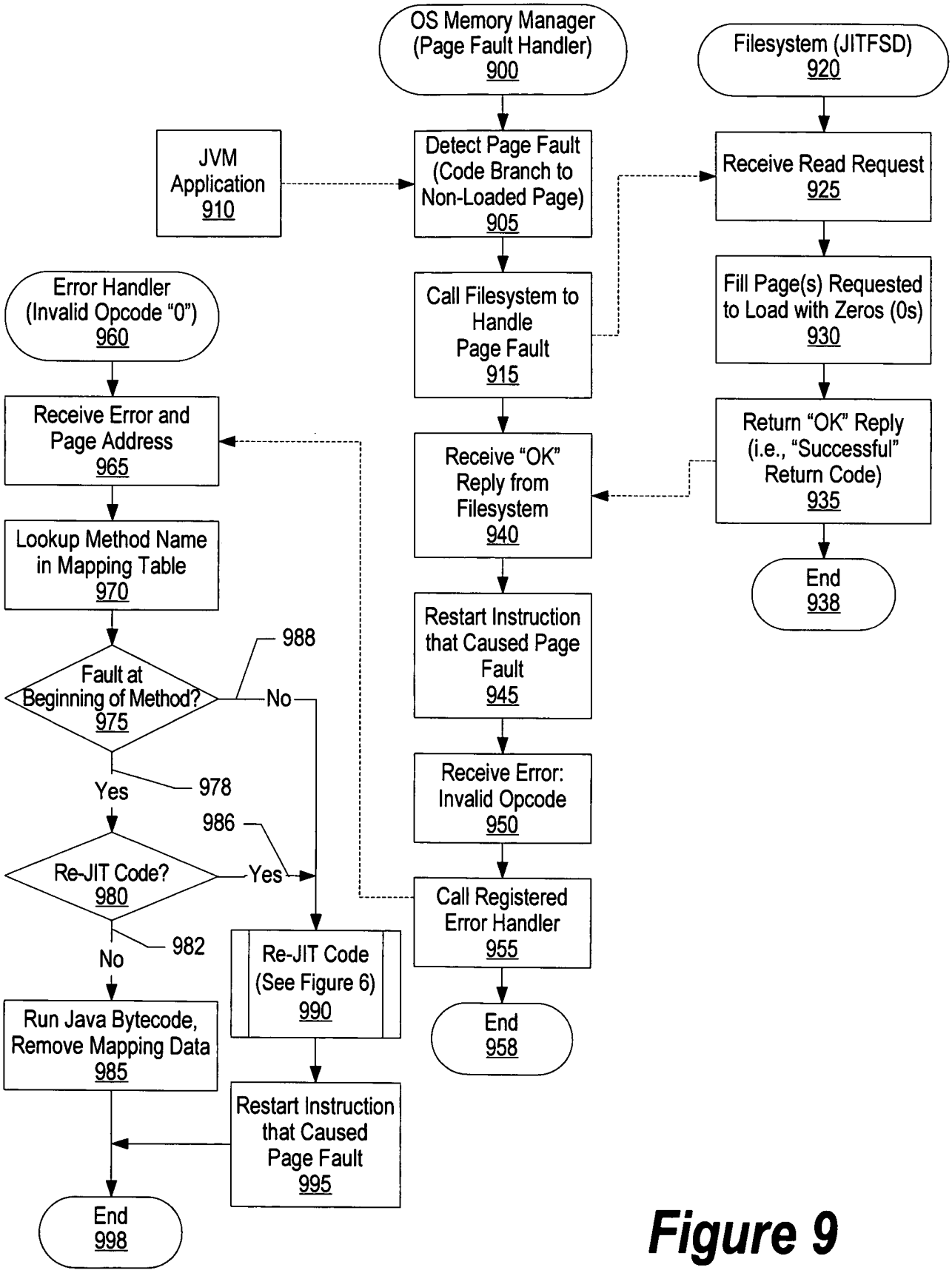
OS Memory Manager (Page Fault Handler) 900
Filesystem (JITFSD) 920
JVM Application 910
Detect Page Fault (Code Branch to Non-Loaded Page) 905
Receive Read Request 925
Call Filesystem to Handle Page Fault 915
Fill Page(s) Requested to Load with Zeros (0s) 930
Error Handler (Invalid Opcode "0") 960
Return "OK" Reply (i.e., "Successful" Return Code) 935
Receive Error and Page Address 965
Receive "OK" Reply from Filesystem 940
End 938
Lookup Method Name in Mapping Table 970
Restart Instruction that Caused Page Fault 945
988
Fault at Beginning of Method? 975
No
978
Yes
Receive Error: Invalid Opcode 950
986
Re-JIT Code? 980
Yes
Call Registered Error Handler 955
982
No
Re-JIT Code (See Figure 6) 990
End 958
Run Java Bytecode, Remove Mapping Data 985
Restart Instruction that Caused Page Fault 995
End 998

Figure 9

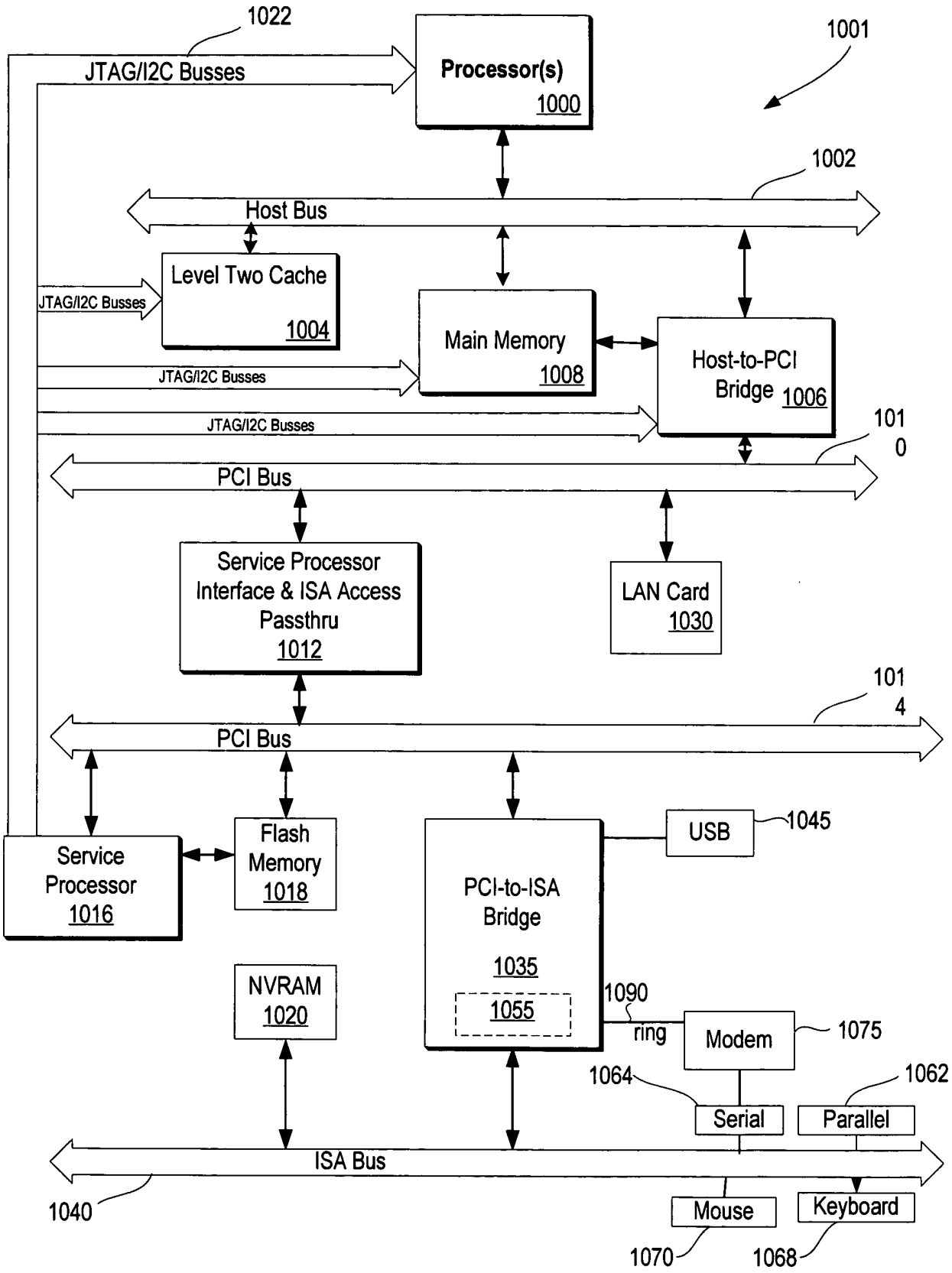
1022
JTAG/I2C Busses
Processor(s)
1000
1001
1002
Host Bus
Level Two Cache
1004
JTAG/I2C Busses
Main Memory
1008
Host-to-PCI
Bridge 1006
JTAG/I2C Busses
JTAG/I2C Busses
1010
PCI Bus
Service Processor
Interface & ISA Access
Passthru
1012
LAN Card
1030
1014
PCI Bus
Service
Processor
1016
Flash
Memory
1018
PCI-to-ISA
Bridge
1035
1055
USB
1045
NVRAM
1020
1090
ring
Modem
1075
1064
Serial
Parallel
1062
ISA Bus
1040
Mouse
Keyboard
1070
1068

Figure 10